AMD

SECURE SYSTEMS DESIGN

LEENDERT VAN DOORN
CORPORATE FELLOW
leendert.vandoorn@amd.com

OUTLINE

- **Introduction**
  - Software Attacks
  - Hardware Attacks
- **Integrity**
  - Immutable Root of Trust
  - Secure Boot
  - Trusted Platform Modules
  - Integrity in a System On a Chip (SOC)
  - Key Distributions Service
- **Isolation**
  - CPU Virtualization
  - I/O Virtualization
  - Secure Coprocessors
  - Encrypted Memory
- **Putting it all together**
  - Secure Hypervisor
  - Physical Secure Processors
- **Summary**

SECURITY IS IN THE NEWS EVERY DAY

DEFINITION: SECURE SYSTEM DESIGN

- Leendert's definition of a secure system

> ***A system that behaves as specified, nothing less, nothing more***

- Some intrinsic challenges
  - Lack of specifications
  - Lack of complete specifications
  - Impossible to prove the absence of *more*
  - Lack of a system view
- Secure system design (like any engineering project) is always a *cost/benefit* tradeoff
- Central to a secure system design are
  - Well-defined security properties (objectives)
  - Threat analysis (what are we protection from whom, cost of entry)
  - Design methodologies (test plan, penetration testing, code review, etc.)
- What happens if you don't do this?

EXAMPLE: CODE INJECTION ATTACK

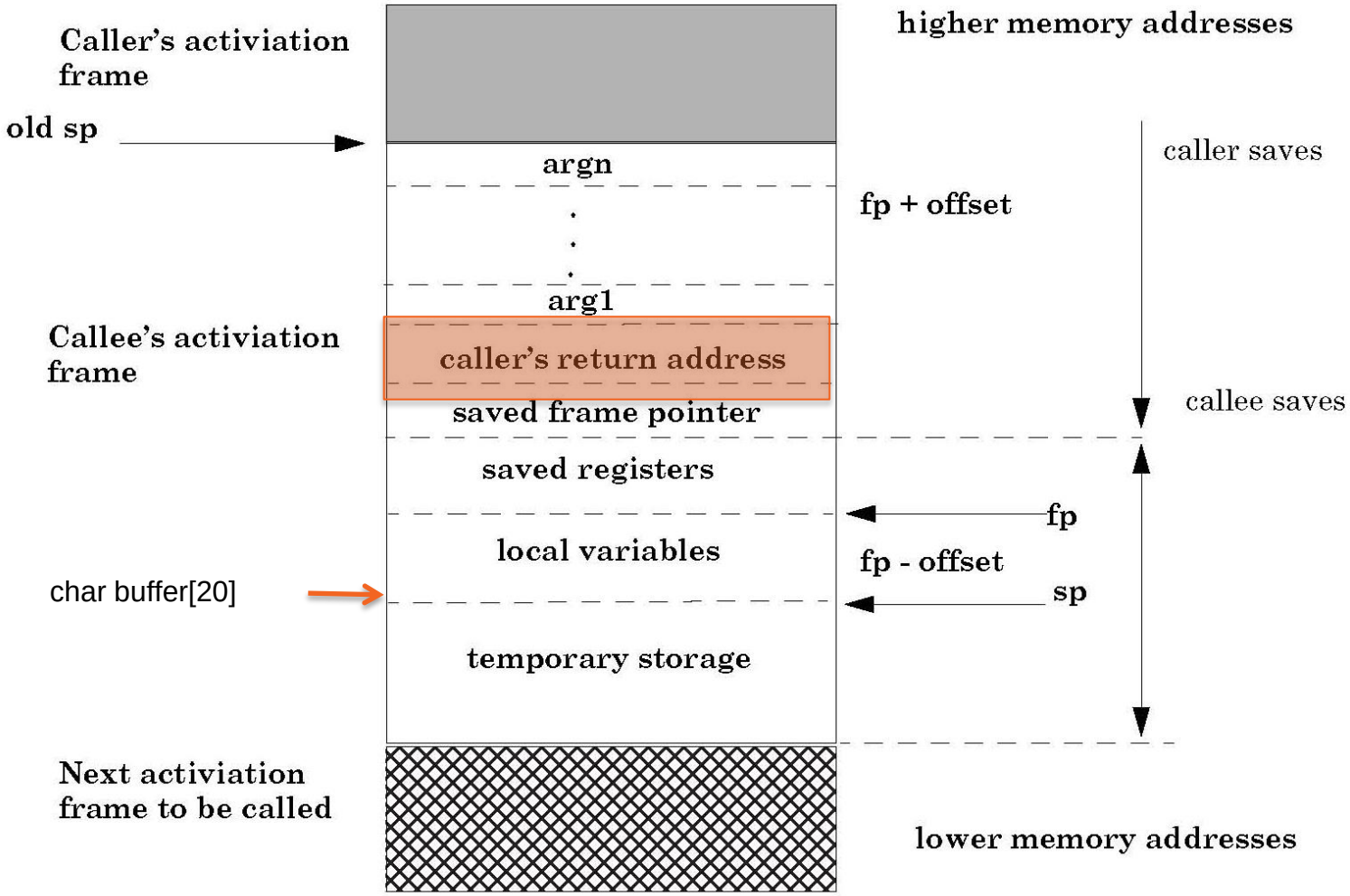
Software is the easiest attack vector

```
void f(void) {
    char buffer[20];
    if (gets(buffer) != NULL)
        ...
```

Input 1:
“0123456789”

Input 2:
“01234567890123456789012345678901234567890123456789...”

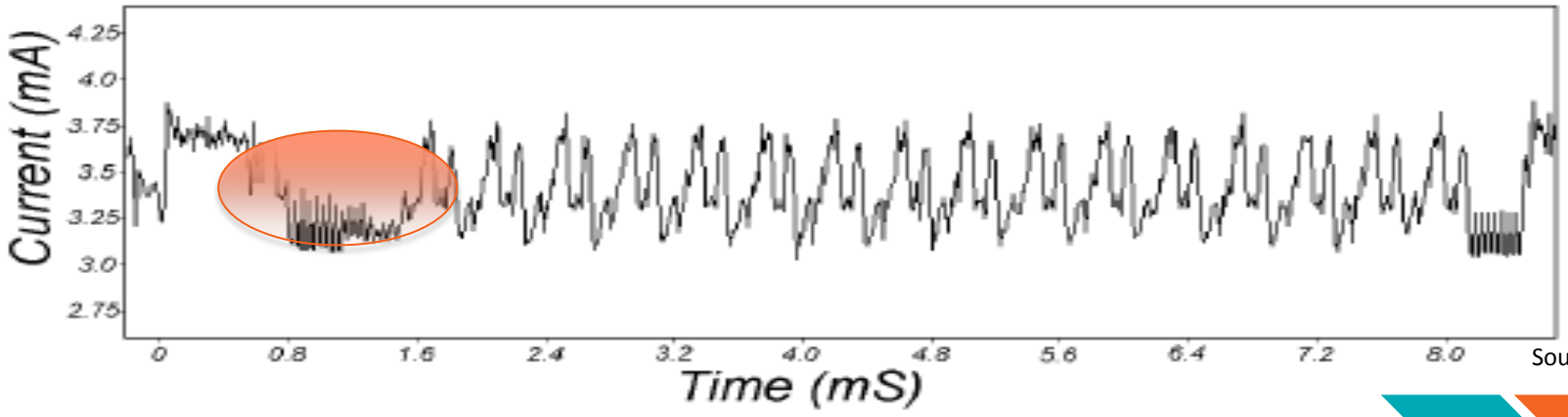
Input 3:
“01234567890123456789...\xR1\xR2\xR3\xR4\x31\xC0...”



EXAMPLE: SIDE CHANNEL ATTACKS

Hardware is not safe either

- Side channels: Electronic components leak a lot of signals
  - Power signals (SPA & DPA)
  - RF signals (EMSEC)
  - Resource contention (cache contention, TLB contention, etc)
  - …
- All these signals can be used to reconstruct the computation at hand
  - Recover secret keys!
- Protecting against these kind of attacks is hard and expensive



As an illustration: Smartcard simple power attack (SPA) where the 16 DES rounds are clearly visible; key schedule computation precedes DES with key clearly visible

Source: Paul Kocher, et. al., Differential Power Analysis

SECURE SYSTEM ELEMENTS

Capabilities typically found in secure systems

AMD

Integrity: Ensuring specified behavior

- Secure boot and Code signing
- Access Control Lists (ACLs) and Capabilities
- Trusted Execution Environments / Secure OSes
- Hardware accelerators (typically crypto, hash, etc)
- Separation of responsibility
- Least privilege
- Defense in depth / layers

Isolation: Eventually things will go wrong

- Minimal Trusted Computing Base (TCB)
- CPU & I/O virtualization
- Dedicated coprocessors
- Enclaves
- Memory encryption
- Hardware tamper resistant
- Shielding (EM leakage, power line filtering, …)

AMD

INTEGRITY

INTEGRITY NEEDS TO BE GROUNDED IN HARDWARE

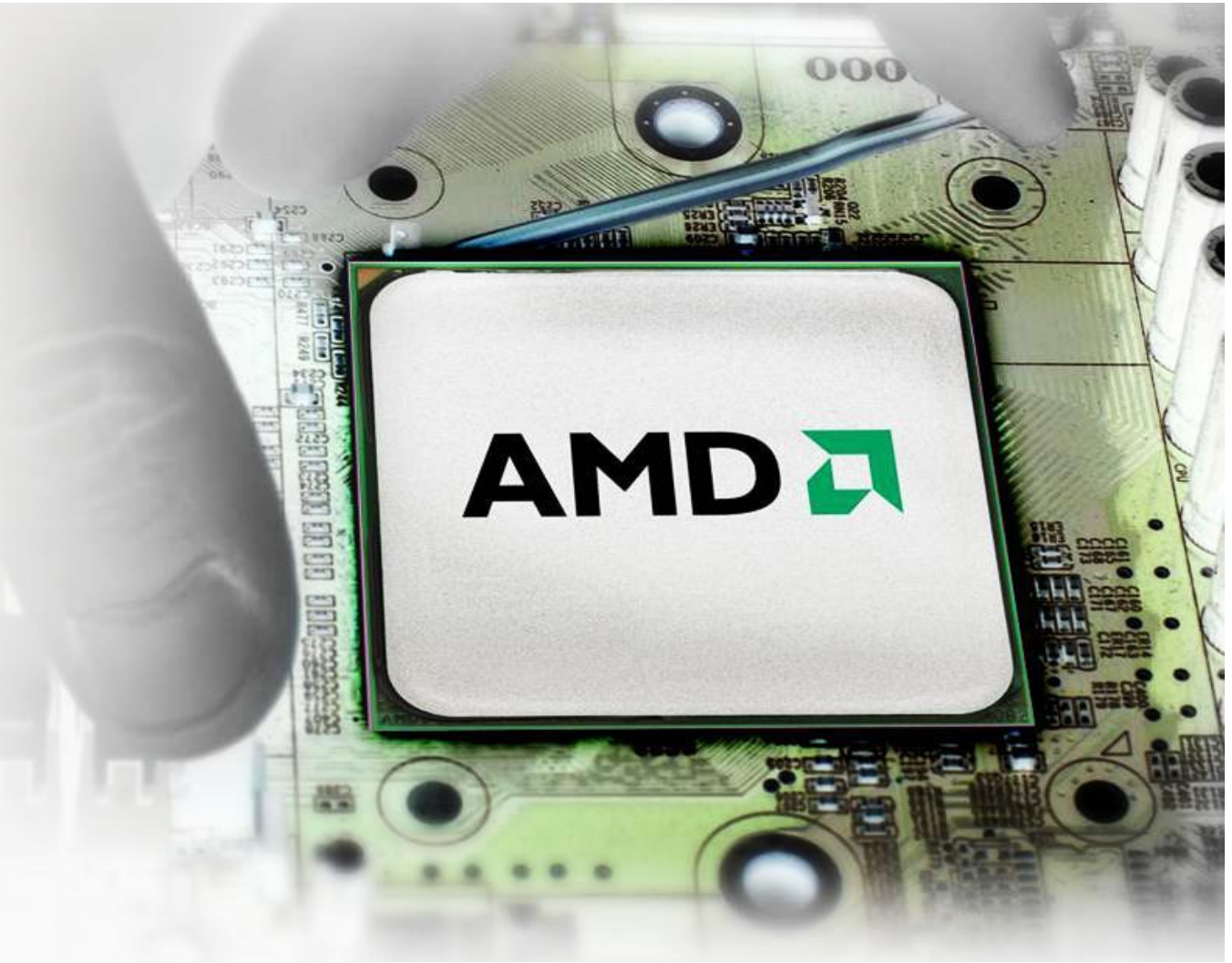
- Integrity starts at the root of a system

  Anything short of that allows an attacker to interpose the bootstrap process and enables BIOS viruses and other Advanced Persistent Threats (APTs).

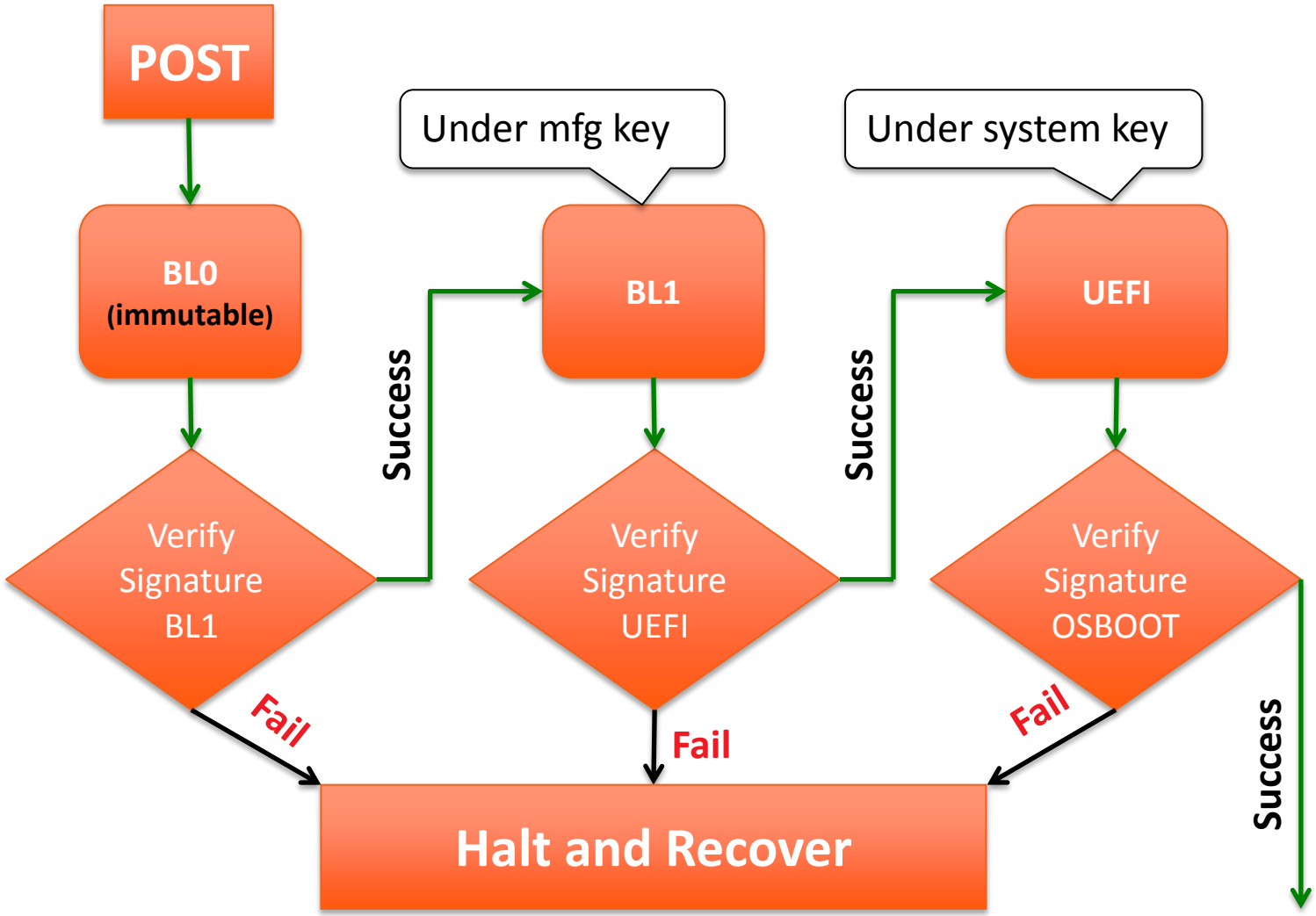
- Integrity needs to be ***anchored*** within the hardware so that it cannot be circumvented
- The root of trust needs to be immutable

Immutable root of trust

- Achievable through read-only boot sectors or separate security processors that guarantee the immutability
  - What guarantees the integrity of the secure processor in that case?



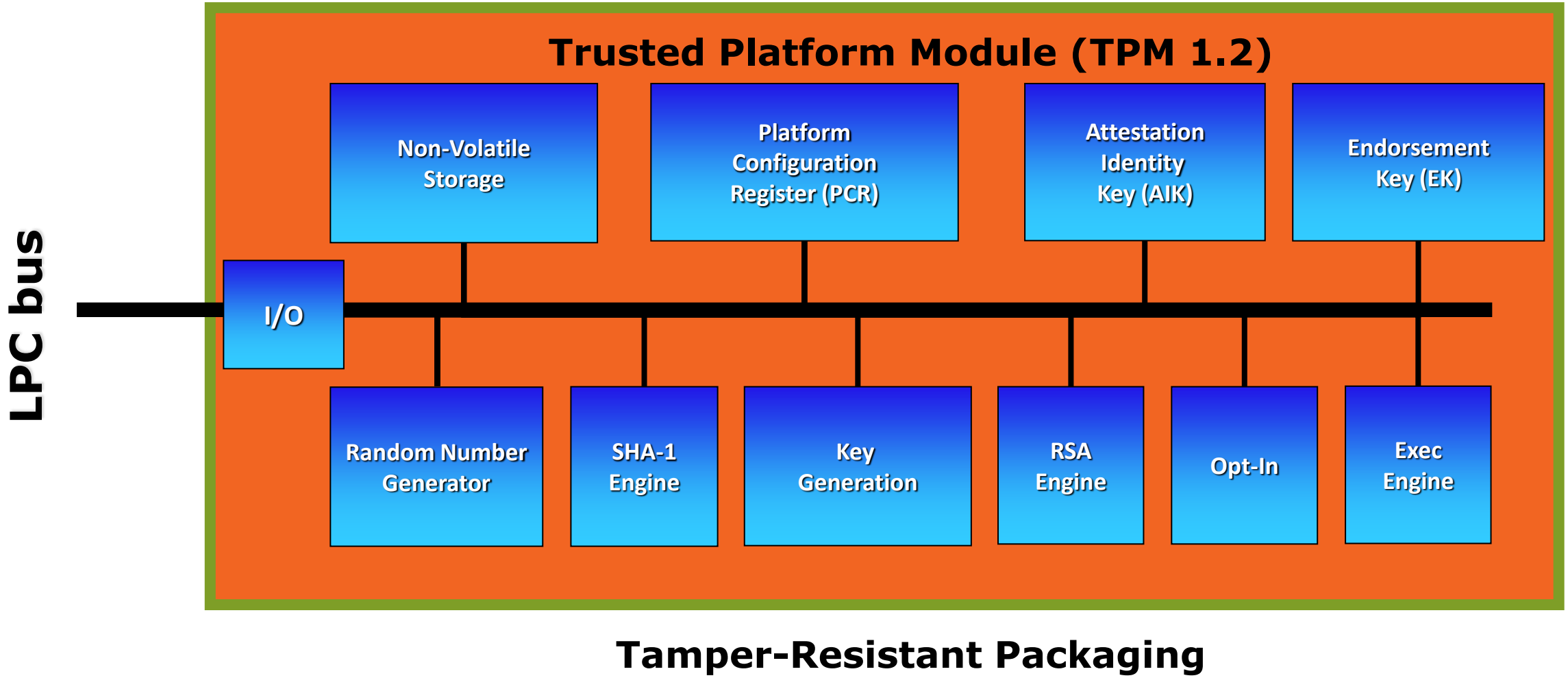
EXAMPLE: SECURE BOOT



You also need to consider:

- Secure key storage for the verification keys
- Roll back prevention for firmware blobs and keys
- Revocation of keys
- Attestation
- Key management

TRUSTED PLATFORM MODULE

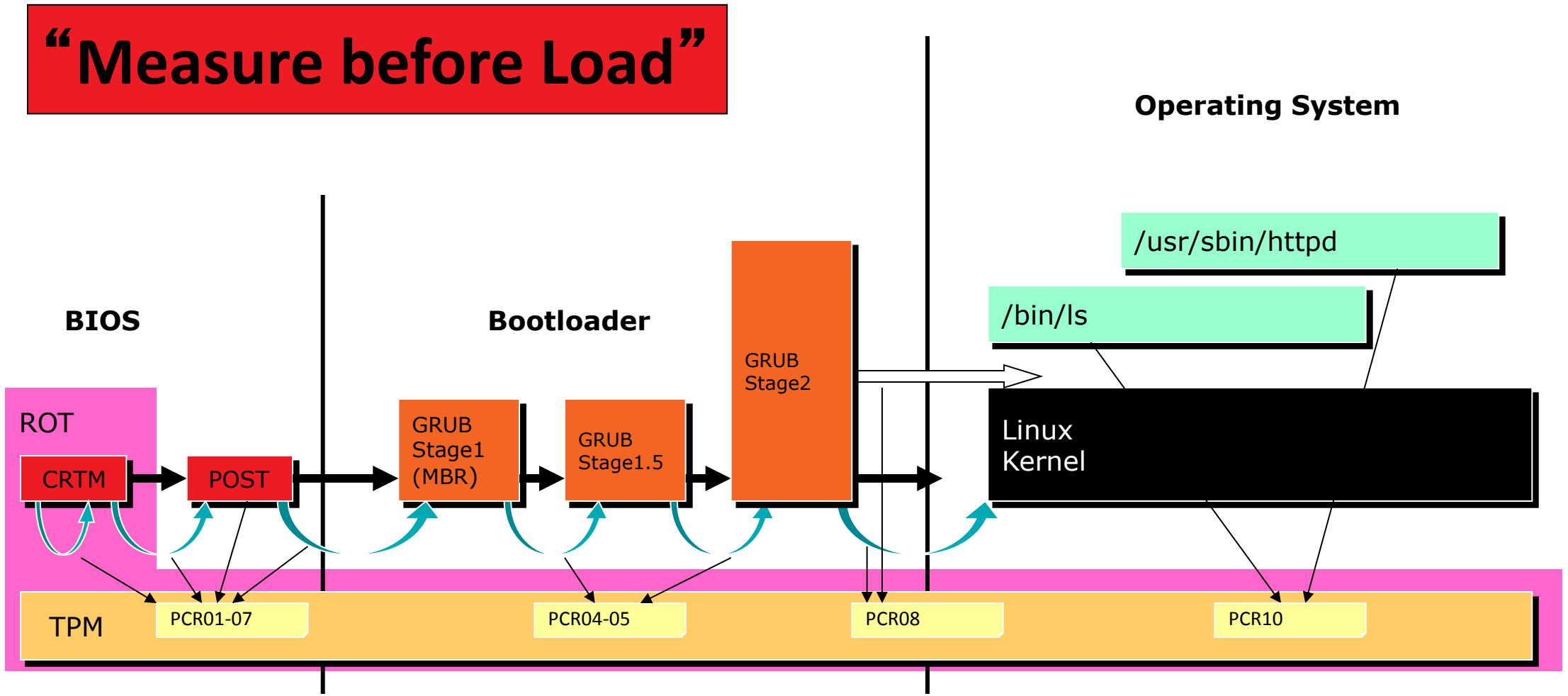


TRUST PLATFORM MODULE PRINCIPLES

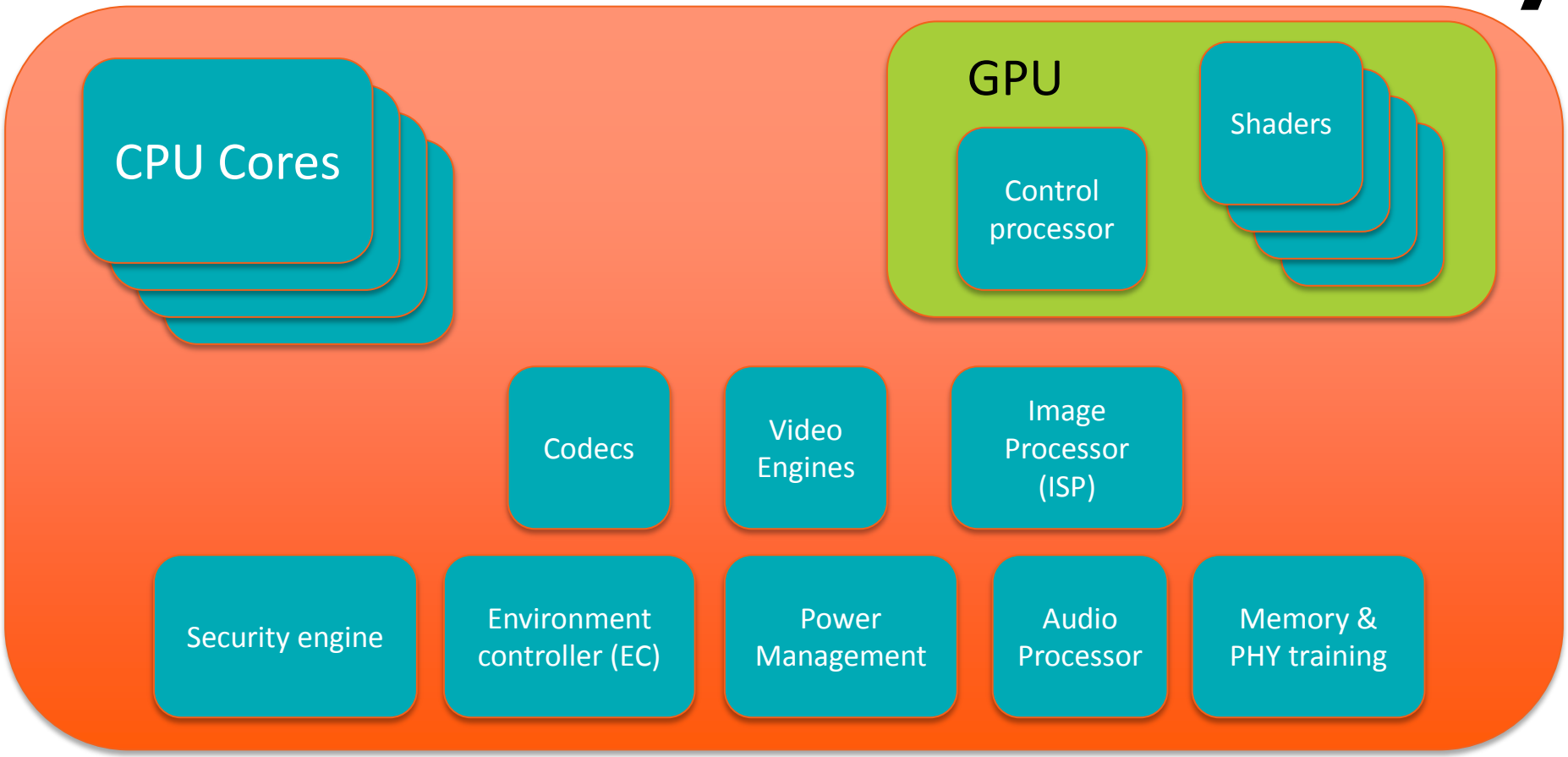
AMD

- Program Configuration Registers (PCR)
  - Extend (PCR_n, H): $PCR_n = f\ (PCR_n\ ||\ H)$, where f is a secure hash function
  - Quote (PCR_n, AIK_i): return PCR_n encrypted under AIK_i
- Sealed Storage
  - Encrypt data under a specific PCR value
  - Can only be released if PCR has that value
- Attestation with help from a trusted 3rd party
- Non-Volatile Storage
- Storage Root Key
- Crypto agility

EXAMPLE: AUTHENTICATED BOOT (PC)



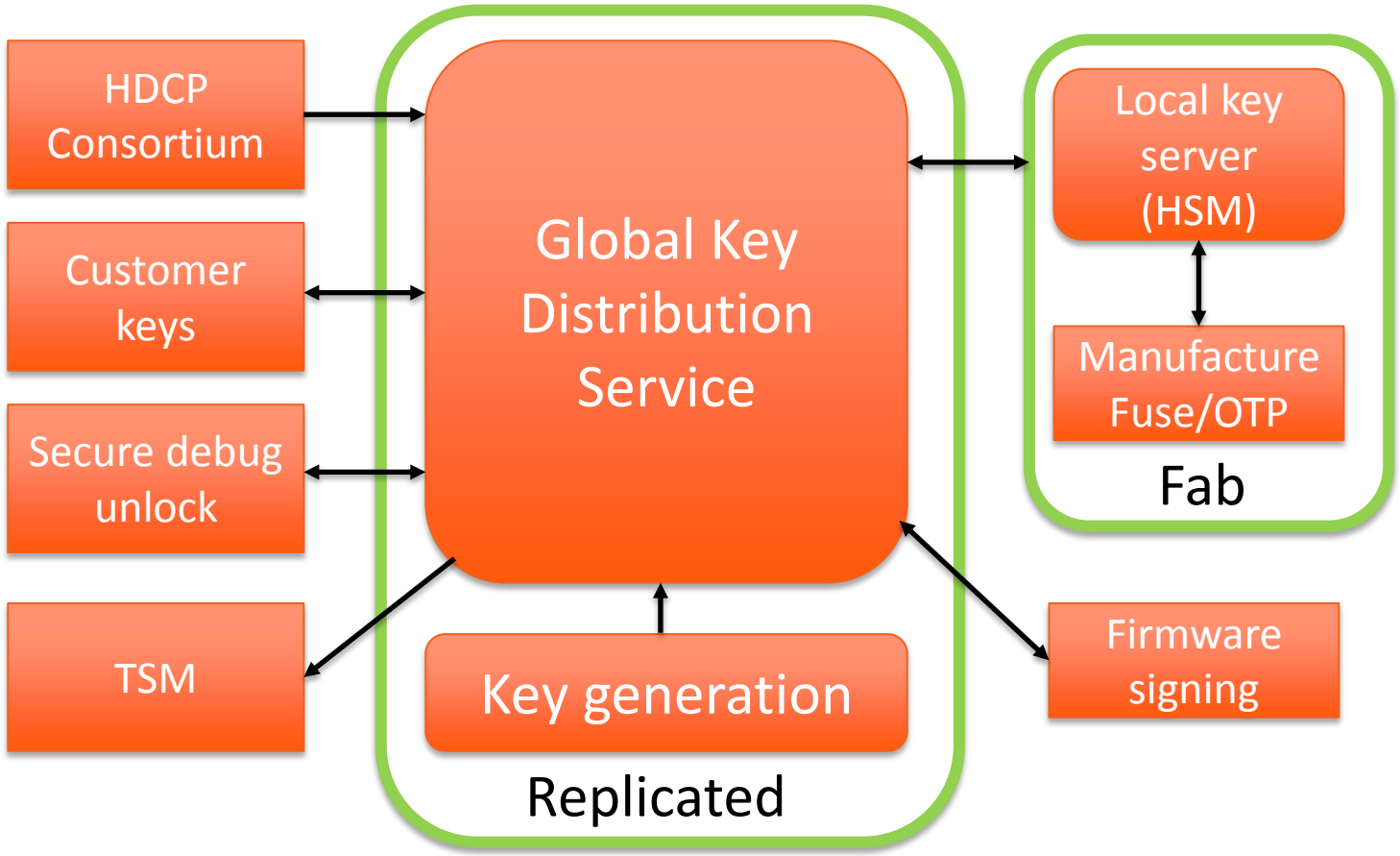
INTEGRITY IN A SYSTEM ON A CHIP (SOC)



- The CPU is no longer the center of an SOC, it is not even the first processor that runs!
- All field-upgradable processors need integrity protection (secure boot)
- What about 3rd party IP?

EXAMPLE: KEY MANAGMENT

- Keys are typically stored in Fuses/OTP
- Typically asymmetric keys
- How to manage these keys?
  - Content protection keys
  - Customer keys
  - Secure debug unlock
  - Trusted Service Managers (TSM)
  - Signed Drivers and firmware
- Fault tolerant design
- Injecting the keys needs to happen in a secure environment
- Secure debug unlock

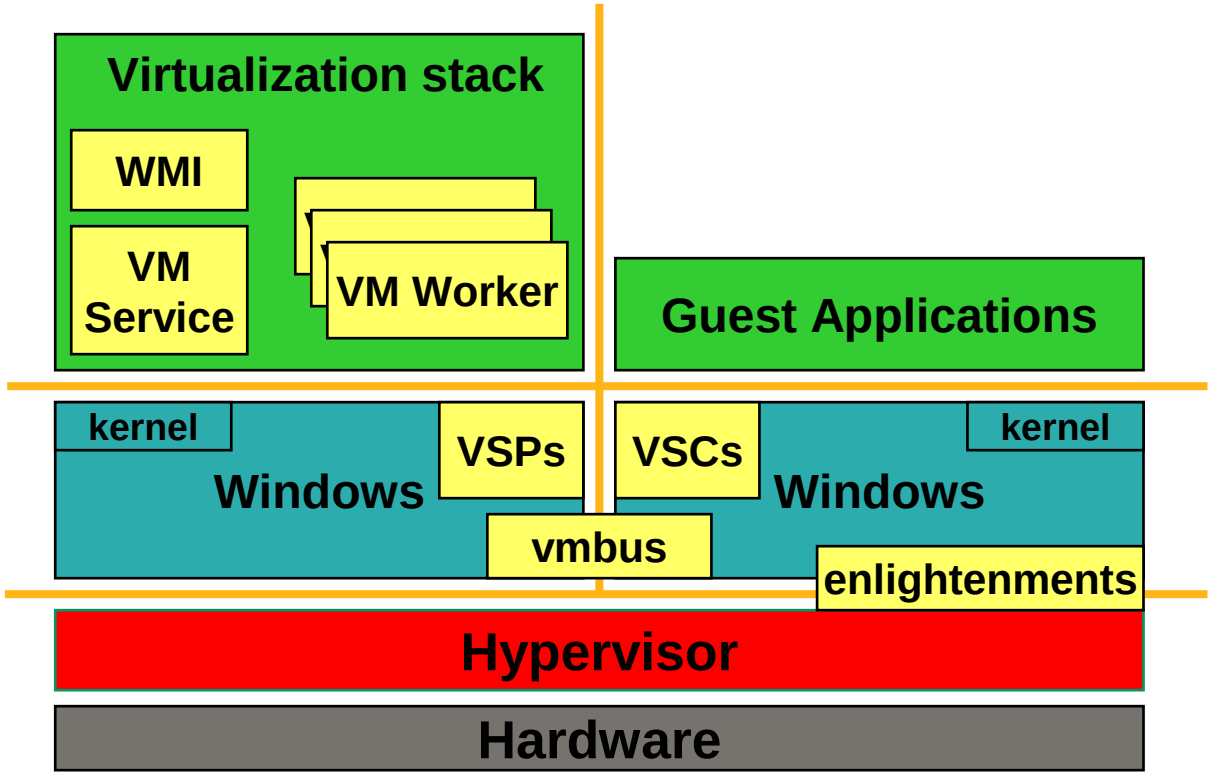


AMD

ISOLATION

CPU VIRTUALIZATION

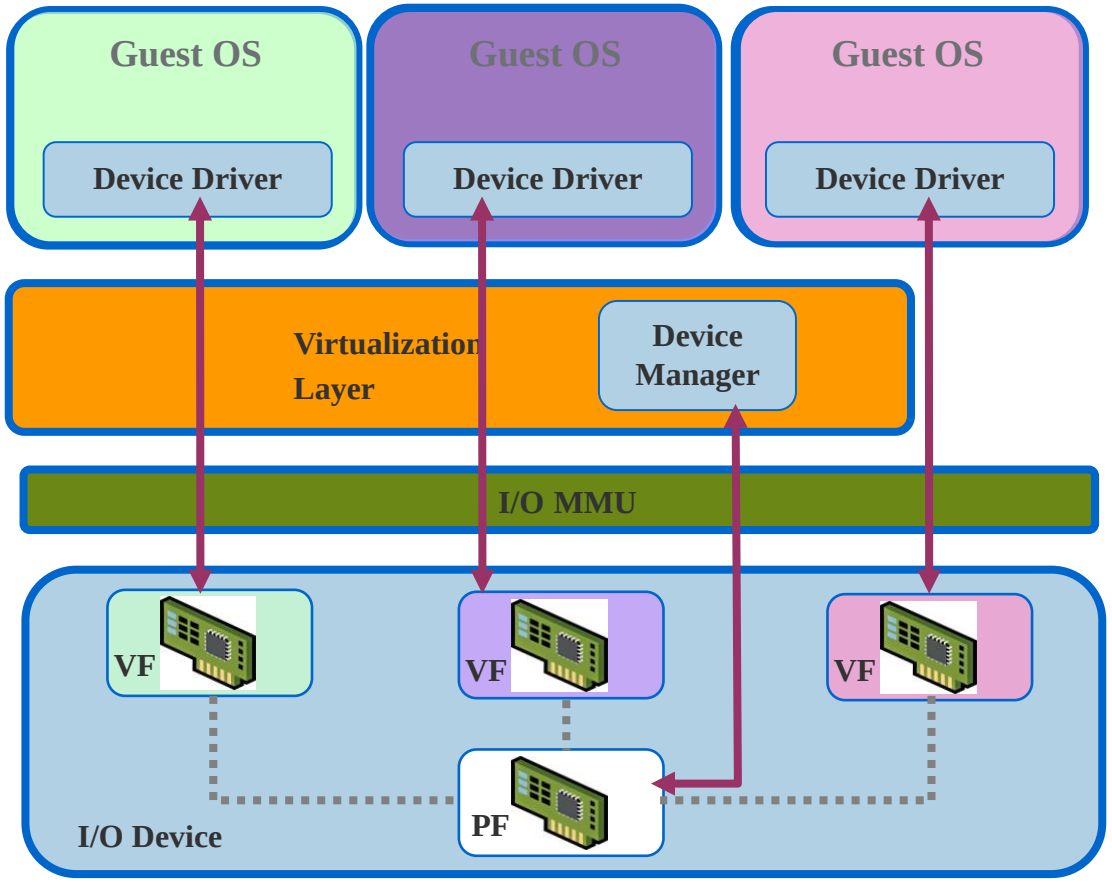
- Multiple consumers share a resource while maintaining the illusion that each consumer owns the full resource
  - Memory, processor(s), storage, peripherals, entire machines
- Goes all the way back to Popek and Goldberg [1974]
- Virtual Machine Monitor (VMM) or hypervisor is the software layer that provides one or more Virtual Machine (VM) abstractions
- Typically used to increase the *utilization* of a system
- Recently also used a *security isolation* mechanism
- Every modern processor (ARM, Intel, AMD) has hardware support for efficient virtualization



Microsoft Hyper-V Hypervisor

I/O VIRTUALIZATION

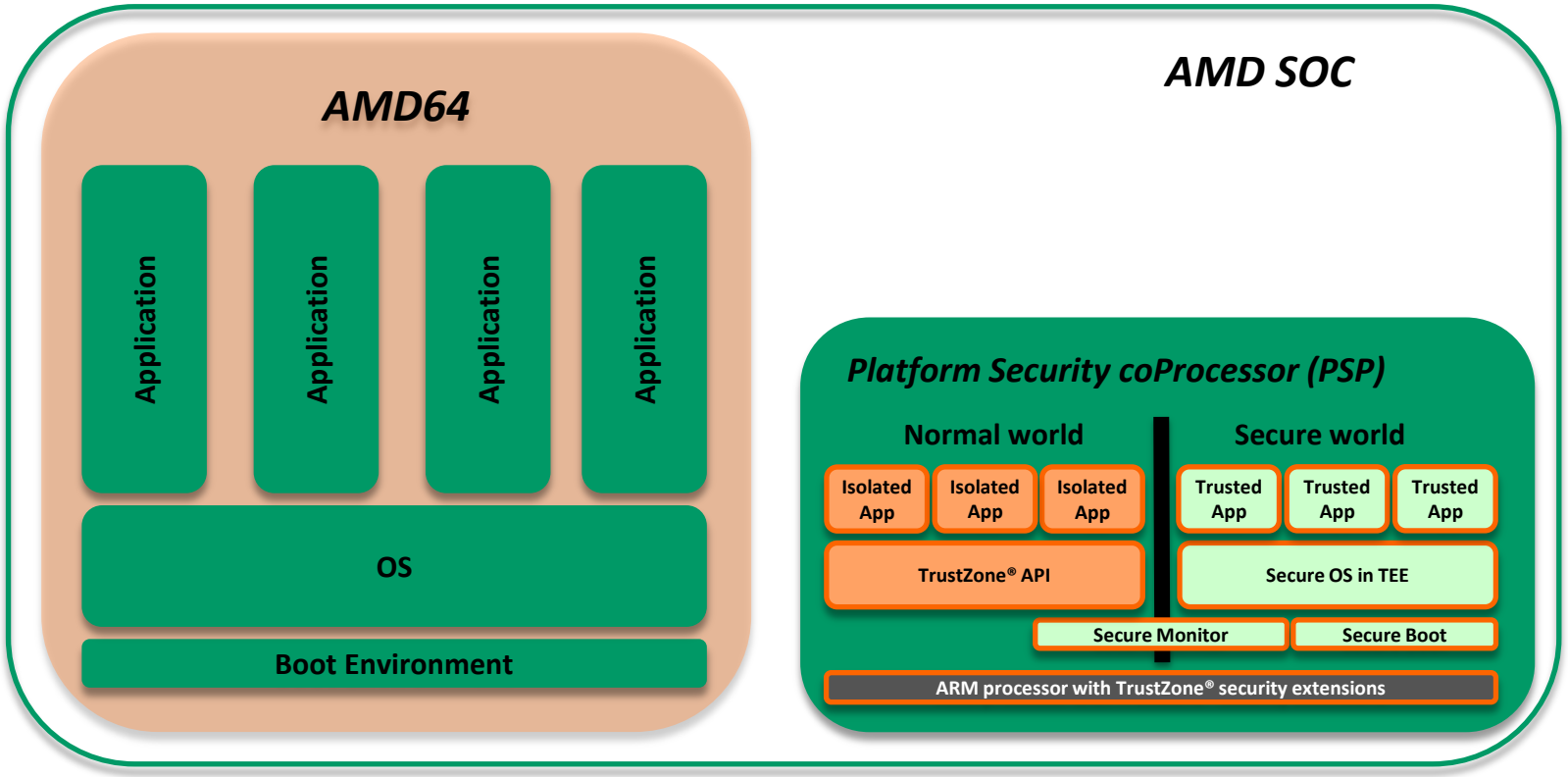
- An IOMMU is akin to an MMU: It translates I/O address to physical memory addresses
- Efficiency: Used to assign I/O (PCIe) devices directly to a virtual machine without going through the hypervisor
- Isolation: Containerizes the damage devices can do through bus master DMA
  - For example USB, Firewire and Display Port attacks
- An IOMMUs is standard in server, in client systems it depends on the value proposition



PF = Physical Function, VF = Virtual Function

Fixed I/O pass-thru with VMWare and an IOMMU

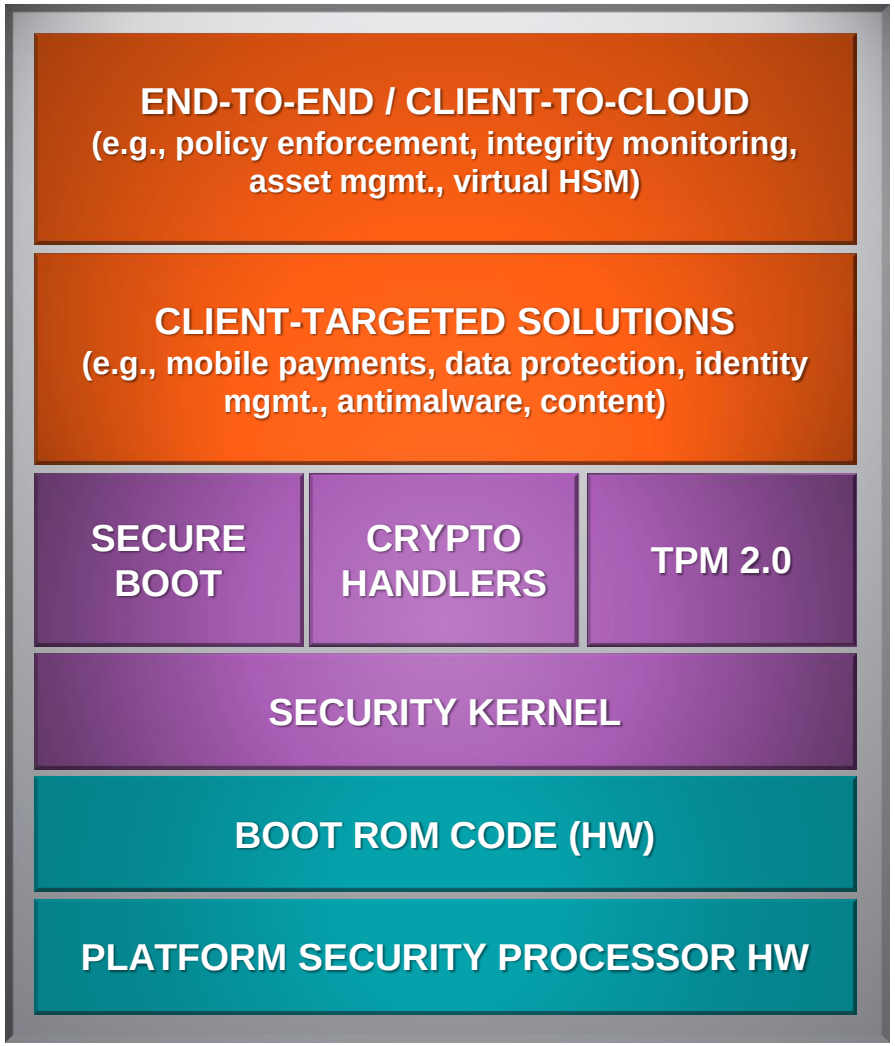
EXAMPLE: AMD'S PLATFORM SECURITY PROCESSOR



- The Platform Security coProcessor (PSP) is an integrated coprocessor next to the AMD64 cores
  - The PSP runs a certified secure OS/kernel
  - The PSP can use Trusted Service Managers (TSM) for provisioning and lifecycle management

PLATFORM SECURITY PROCESSOR APPLICATIONS

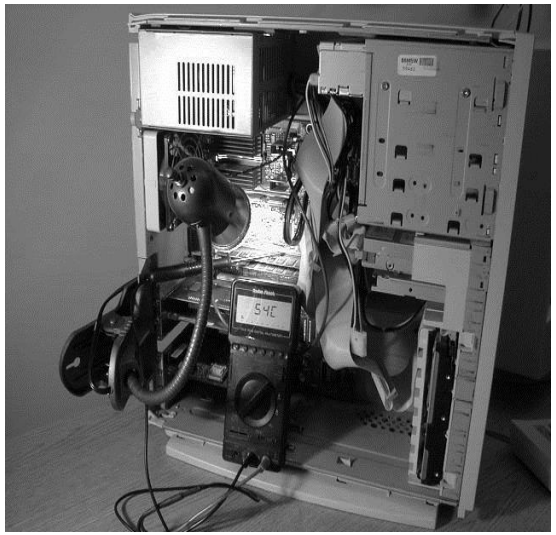
- Platform Security Foundational support
  - Trusted Execution Environment
  - Secure boot
  - Cryptographic acceleration
  - TPM functionality
- Client solutions enablement
  - 3rd party solutions – e.g., payments, anti-theft, identity management, data protection, anti-malware, content protection, bring-your-own-device
- End-to-end / client-to-cloud
  - 3rd party solutions – e.g., vertical solutions, policy enforcement, integrity monitoring, audit & asset management, virtual HSM



ENCRYPTED MEMORY: THE SOC AS TRUST BOUNDARY

- Generally, all software code and data is stored in DRAM external to the SOC
  - An attacker with physical access to the DRAM could
  - Observe secrets (like encryption keys)
  - Re-write code (bypass authentication checks)
  - Etc.
- Well-known memory attacks include
  - HW Probing/monitoring
    - Probing the physical DRAM interface to observe data going to the SOC
  - DMA Attacks
    - Using a malicious hardware connection to read/write DRAM directly
    - Examples: Firewire/Thunderbolt attacks
  - Cold Boot
    - Freezing a DRAM chip and transferring it to another computer to read its data
    - Potential future NV technologies (e.g. memristor) would not even require freezing
  - Induce memory errors and exploit those

Source: https://citp.princeton.edu/research/memory/



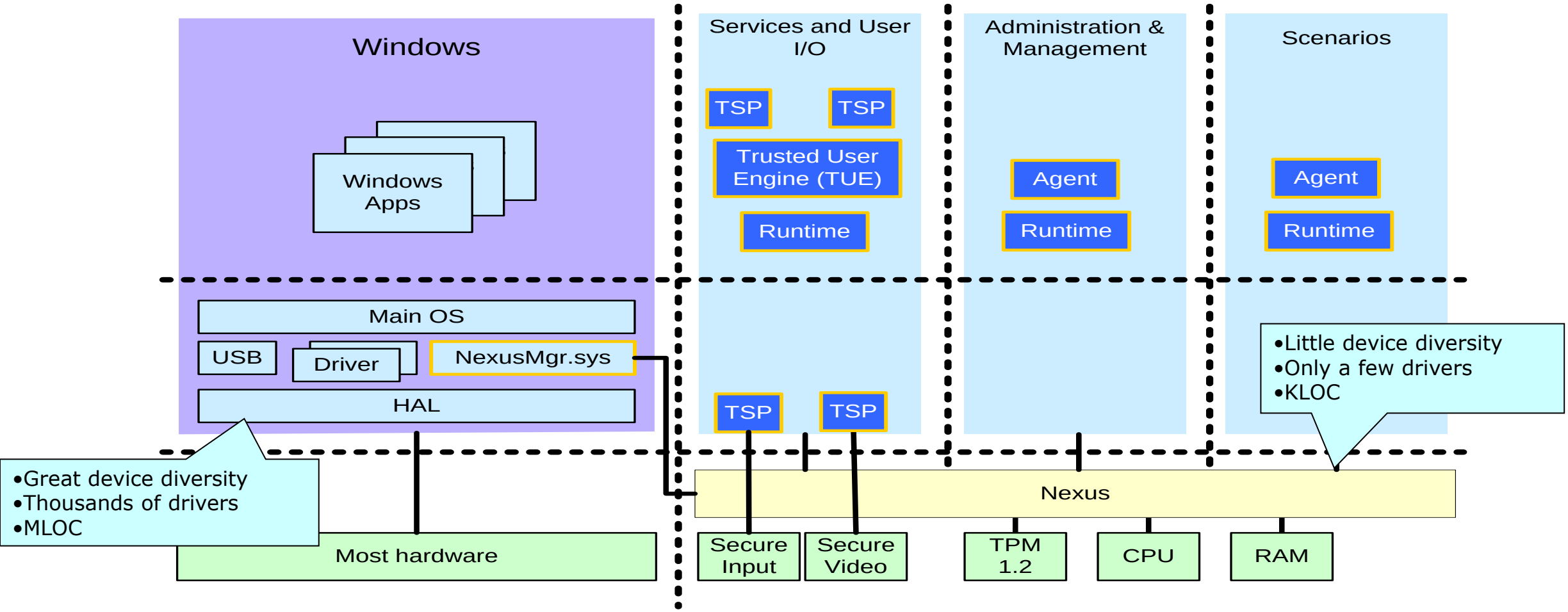
Source: https://www.cs.princeton.edu/~appel/papers/memerr.pdf

EXAMPLE: ENCRYPTED MEMORY IMPLEMENTATION

- Attack model: Everything outside the SOC is untrusted
  - Memory modification attacks
  - Rollback attacks
  - Cut & paste attacks
- Solution: All memory is encrypted and integrity protected
- Efficient implementation is possible using counter modes such as
  - IAPM, OCB and GCM (Galois/Counter Mode, part of suite-B)
  - Integrate both confidentially and integrity in a single pass
  - Curry-in memory locations and/or generation counters, or IV
- Where to implement memory encryption?
  - Cache controller
  - North Bridge
  - Memory controller
- Challenges
  - Increases memory access latencies
  - SRAM required to keep the state (typically solved by partial integrity guarantees)

PUTTING IT ALL TOGETHER

EXAMPLE: SECURE SYSTEM (MICROSOFT'S NGSCB 2004)



Source: Microsoft's WinHEC 2004 presentation

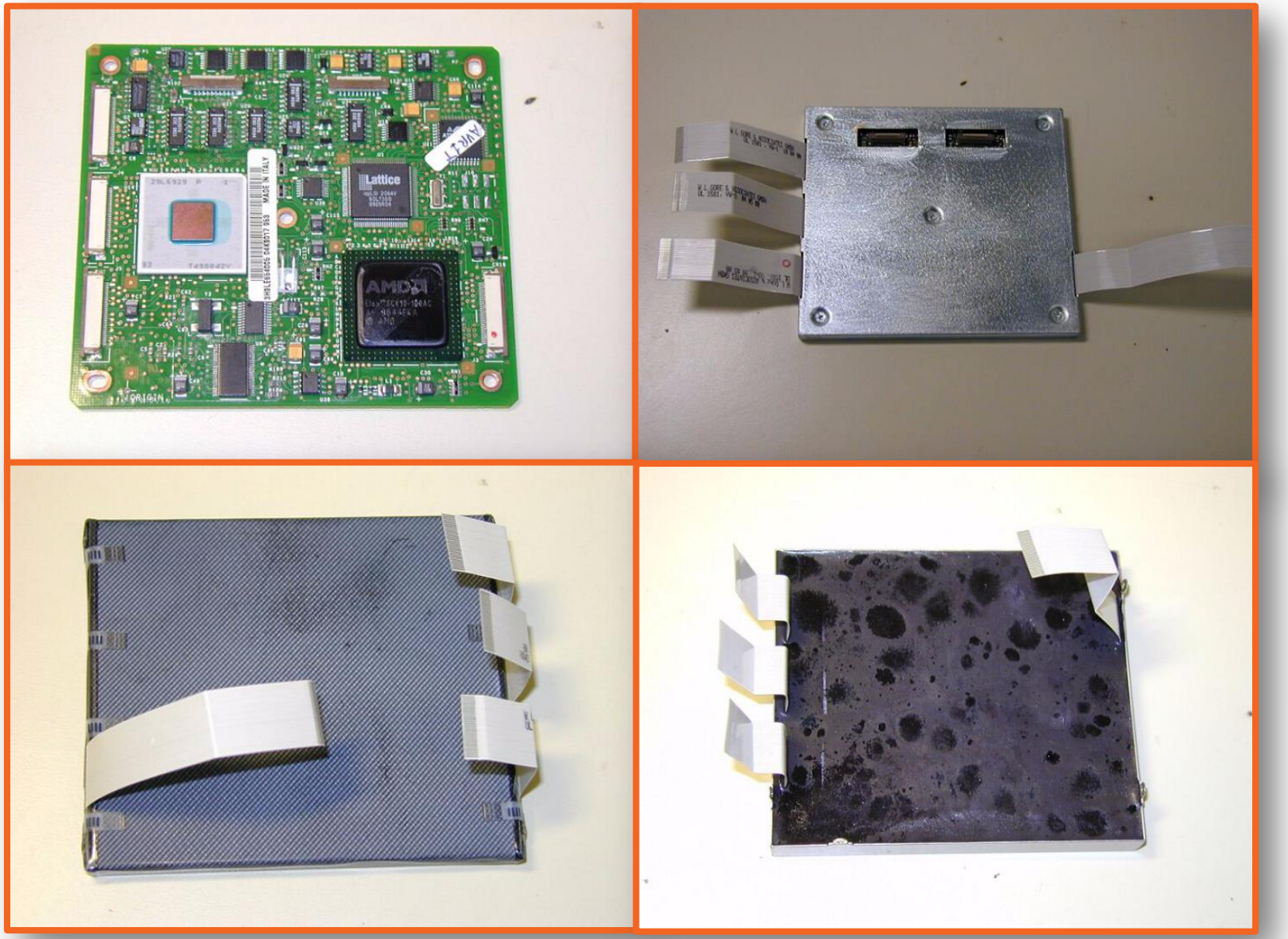
EXAMPLE: IBM'S SECURE CRYPTOGRAPHICS COPROCESSORS

IBM 47xx PCI Cryptographic Coprocessor

- High-speed cryptography (for back then)
- Provides secure storage (e.g., keys)
- Tamper-resistant, sensing and responding
  - Detecting physical attacks: probe, voltage, temperature, radiation
- Programmable
- Secure configuration and field updates
- Supported in PCs → mainframes
  - Windows, Solaris, Linux, AIX, OS/390, OS/400
- 4758 in 2001, followed in 2003 by the 4764, and in 2010 the 4765

Source: http://www-03.ibm.com/security/cryptocards/pcixcc/overhardware.shtml

4758 SECURE PACKAGING



Source: Ron Perez, IBM T.J. Watson Research Center

SUMMARY

AMD

- Attacks
  - Software (easiest attack vector today)
  - Hardware (not hard but requires physical access)
  - The number of hardware attacks are rising
- Two key building blocks
  - Integrity (know what it is you are running)
  - Isolation / containment (prevent things going wrong and control them when they do)

MATERIAL FOR FURTHER STUDY

AMD

- Secure system design
  - Ross Anderson, Security Engineering
- Side channel attacks
  - Jude Ambrose, Power Analysis Side Channel Attacks: The Processor Design-level Context
  - Paul Kocher, et al., Differential Power Analysis
- Memory attacks
  - Princeton memory attacks: https://citp.princeton.edu/research/memory/, https://www.cs.princeton.edu/~appel/papers/memerr.pdf
  - USB, Firewire & Display Port DMA attacks: http://work.delaat.net/rp/2011-2012/p14/report.pdf
- Code inject attacks
  - Roemer, et al., Return oriented programming: Systems, Languages, and Applications
- Secure boot
  - Bill Arbaugh, *A secure and reliable bootstrap architecture*
- TCG & TPM
  - http://www.trustedcomputinggroup.org
  - Sailer et al. Design and Implementation of a TCG-based Integrity Measurement Architecture
- Secure virtualization
  - Ron Perez, et al., Virtualization and Hardware-Based Security
- I/O virtualization
  - Muli-Ben Yehuda, Utilizing IOMMUs for Virtualization in Linux and Xen
- Trusted Execution Environments
  - Global Platform
- Secure Coprocessor
  - Todd Arnold, et al.,The IBM PCIXCC: A new cryptographic coprocessor for the IBM eServer

GLOSSARY OF TECHNICAL TERMS

ACL – Access Control List

AIK – Attestation Identity Key

CPU – Central Processing Unit

CRTM - Core Root of Trust Management

DES – Data Encryption Standard (obsolete algorithm)

DPA – Differential Power Analysis

EM – Emission

EMSEC - Emission Security

GPU – Graphics Processing Unit

GRUB – Linux bootstrap loader

HSM – Hardware Security Model

IOMMU – I/O Memory Management Unit

MMU - Memory Management Unit

OTP – One Time Programmable

PCR – Program Configuration Register

POST – Power On Self Test

ROT – Root of Trust

SOC – System On a Chip

SPA – Simple Power Analysis

TCB – Trusted Computing Base

TLB – Translations Lookaside Buffer

TPM – Trusted Platform Module

TSM – Trusted Service Manager

VM – Virtual Machine

VMM – Virtual Machine Monitor

BIOGRAPHY

Dr. **Leendert van Doorn** is a Corporate Fellow and Corporate VP at AMD where he is responsible for driving software innovation across the company. He actively engages with AMD's software partners to understand their long-term roadmaps and reflects this input back into AMD's roadmaps, while at the same time evangelizing AMD's future plans with AMD's partners.

Leendert is responsible for AMD's security, virtualization, manageability, and software ecosystem strategies. During the last 4 years he has been actively driving AMD's ARM 64-bit server ecosystem enablement. He is a member of AMD's Innovation Leadership Team and actively participates in AMD's domain roadmap process.

Before joining AMD he was a Sr. Manager at IBM's T.J. Watson Research Center where he lead virtualization, system security, penetration and security usability research teams. Leendert holds a Ph.D. from the *Vrije Universiteit* in Amsterdam, The Netherlands. Occasionally he is known to find refuge at Carnegie Mellon University where he is an adjunct professor.

Trademark Attribution

AMD, the AMD Arrow logo and combinations thereof are trademarks of Advanced Micro Devices, Inc. in the United States and/or other jurisdictions. Other names used in this presentation are for identification purposes only and may be trademarks of their respective owners.